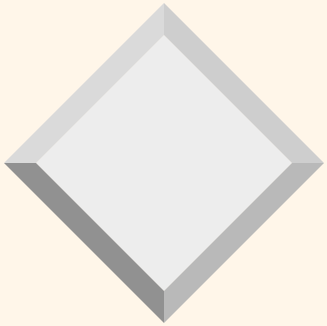
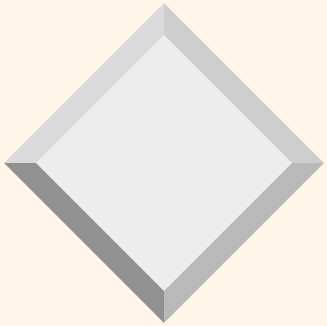
Prelim discussion

- A sample (from last Spring) is on CMS
- A 12-hour take-home
  - Noon to midnight
  - Should take *much* less than 12 hours
- Two (different) exams
  - Thurs 13 Oct
  - Sun 16 Oct
- Email me *real soon now* if you have problems with both days



Database Design



Readings: [RG] Sec. 2.1-2.5, 3.5

Part 3 of the course – DB design

- How to **apply** the relational DB abstraction to a real world problem?
- How do I create a schema that **matches reality** and has **good mathematical properties** (e.g. no redundancy?)
  - A lot of what we do applies to nonrelational abstractions too
- Shorter unit (about 1 week)
  - H3 material

Part 3 of the course – DB design

- Designing a database is a complex, multi-step process
  - And often iterative
  - Once you finally "get it right", real-world scenario changes and you have to start again!

The Database Design Process

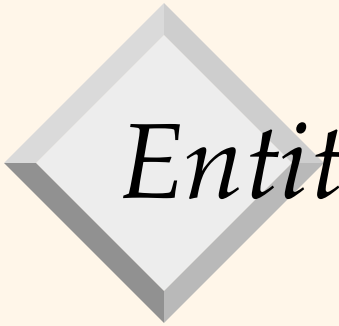
- Requirements analysis
  - based on use cases, description of business processes, etc.
- Conceptual design (create data model)
  - What is the DB actually *about*?
  - May use tools such as the ER model
- Schema refinement/normalization
  - Does the schema have any "bad properties" (e.g. redundancy) that should be removed?
- Physical tuning
  - Based on factors like known query workload
  - Choose indexes etc.

The Database Design Process

- Requirements analysis
  - based on use cases, description of business processes, etc.
- Conceptual design (create data model)
  - What is the DB actually *about*?
  - May use tools such as the ER model
- Schema refinement/normalization
  - Does the schema have any "bad properties" (e.g. redundancy) that should be removed?
- Physical tuning
  - Based on factors like known query workload
  - Choose indexes etc.

Conceptual design

- What are the entities and relationships in the real world?
- What information about these entities and relationships should we store in the database?
- What are the integrity constraints or business rules that hold?
- Handy modeling tool: ER (entity-relationship) diagrams



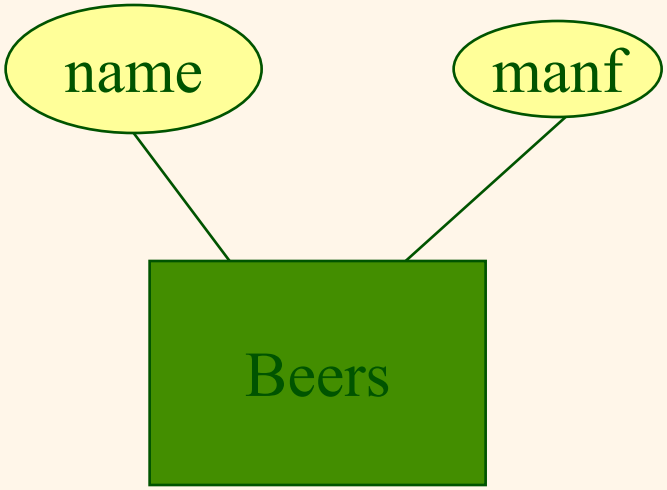
Entity Sets

- *Entity* = "thing" or object.
- *Entity set* = collection of similar entities.
  - Similar to a class in OO languages
  - **Instance of an entity set** = set of actual entities
- *Attribute* = property of (the entities of) an entity set.
  - Attributes are simple values, e.g. integers or character strings, not structs, sets, etc.

ER Diagrams

- In an entity-relationship diagram:
  - Entity set = rectangle.
  - Attribute = oval, with a line to the rectangle representing its entity set.

Example:

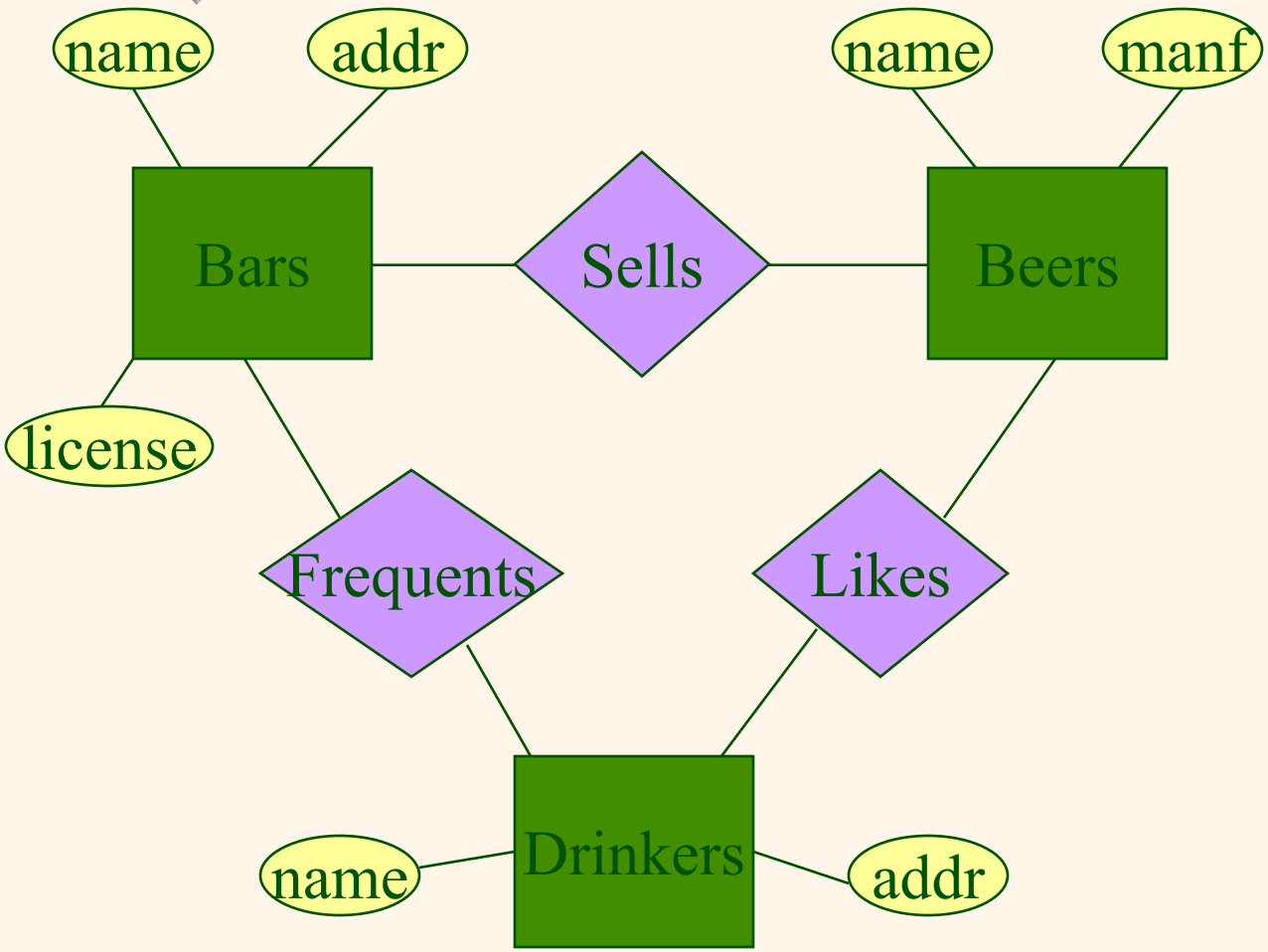


- Entity set Beers has two attributes, name and manf (manufacturer).
- Each Beers entity has values for these two attributes, e.g. (Bud, Anheuser-Busch)

Relationships

- A **relationship** connects two or more entities.
- Analogous to entity sets, there is a notion of **relationship sets.**
- A relationship set is represented by a diamond, with lines to each of the entity sets involved.

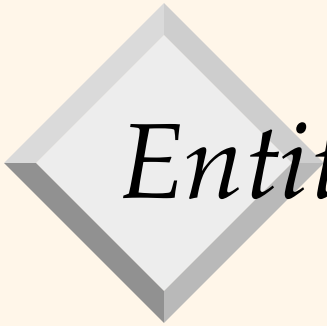
Example



Bars sell some beers.

Drinkers like some beers.

Drinkers frequent some bars.



Entity Sets, Relationship Sets

- The current "value" of an entity set is the set of entities that belong to it.
  - Example: the set of all bars in our database.
  - *Instance* of an entity set

Entity Sets, Relationship Sets

- The "value" of a relationship is a *relationship set*, a set of tuples with one component for each related entity set
  - *Instance* of a relationship set
  - Two entities can only participate in each relationship together once (set, not multiset)

Example: Relationship Set Instance

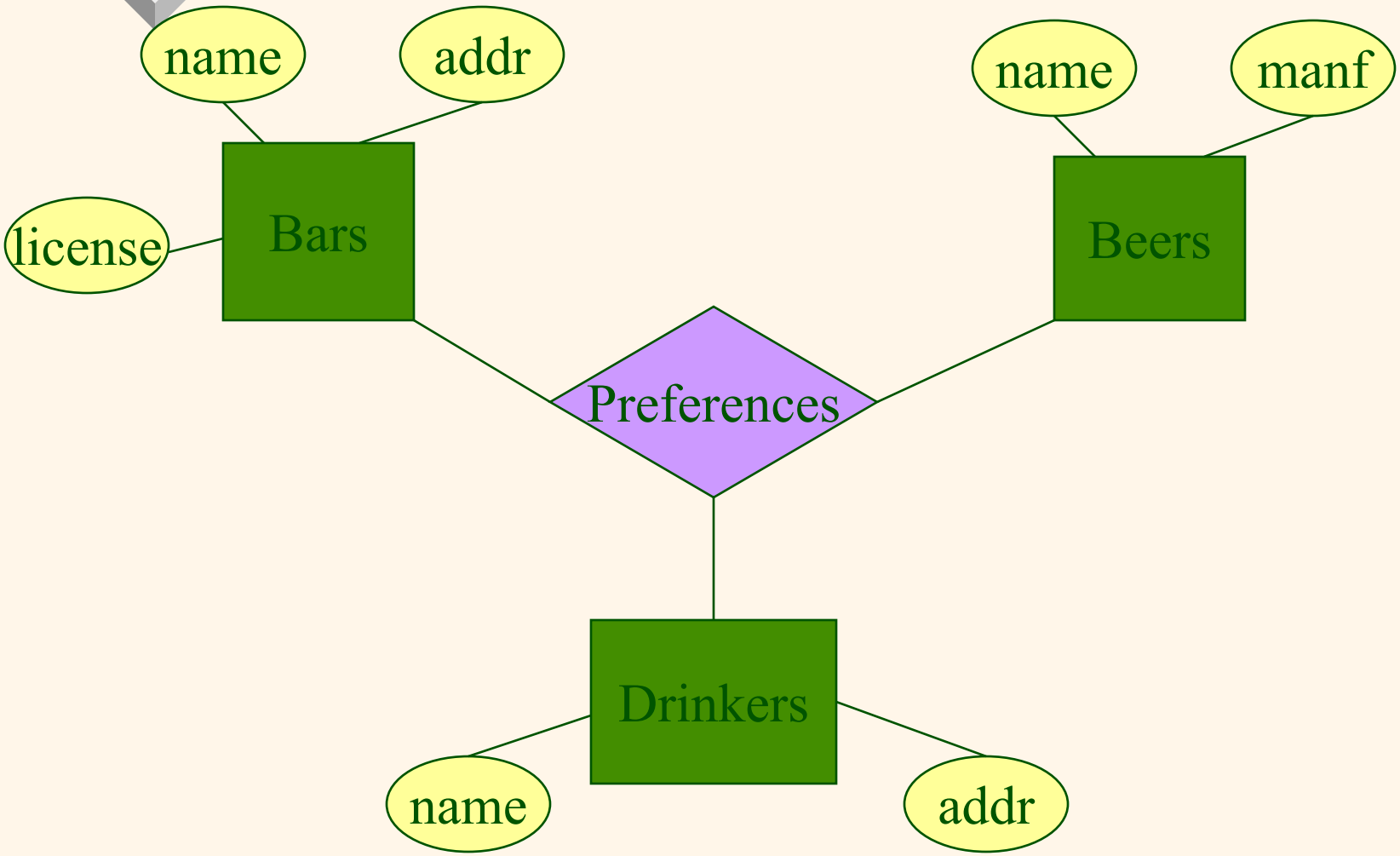
- For Sells, we might have a relationship set instance like this:
  - Notice no duplicates (since it's a *set*)

bar	beer
Joe's Bar	Bud
Joe's Bar	Miller
Sue's Bar	Bud
Sue's Bar	Pete's Ale
Sue's Bar	Heineken

Multiway Relationships

- Sometimes, we need a relationship that connects more than two entity sets.
- Suppose that drinkers will only drink certain beers at certain bars.
  - Our three binary relationships Likes, Sells, and Frequents do not allow us to make this distinction.
  - But a 3-way relationship would.

Example: 3-Way Relationship



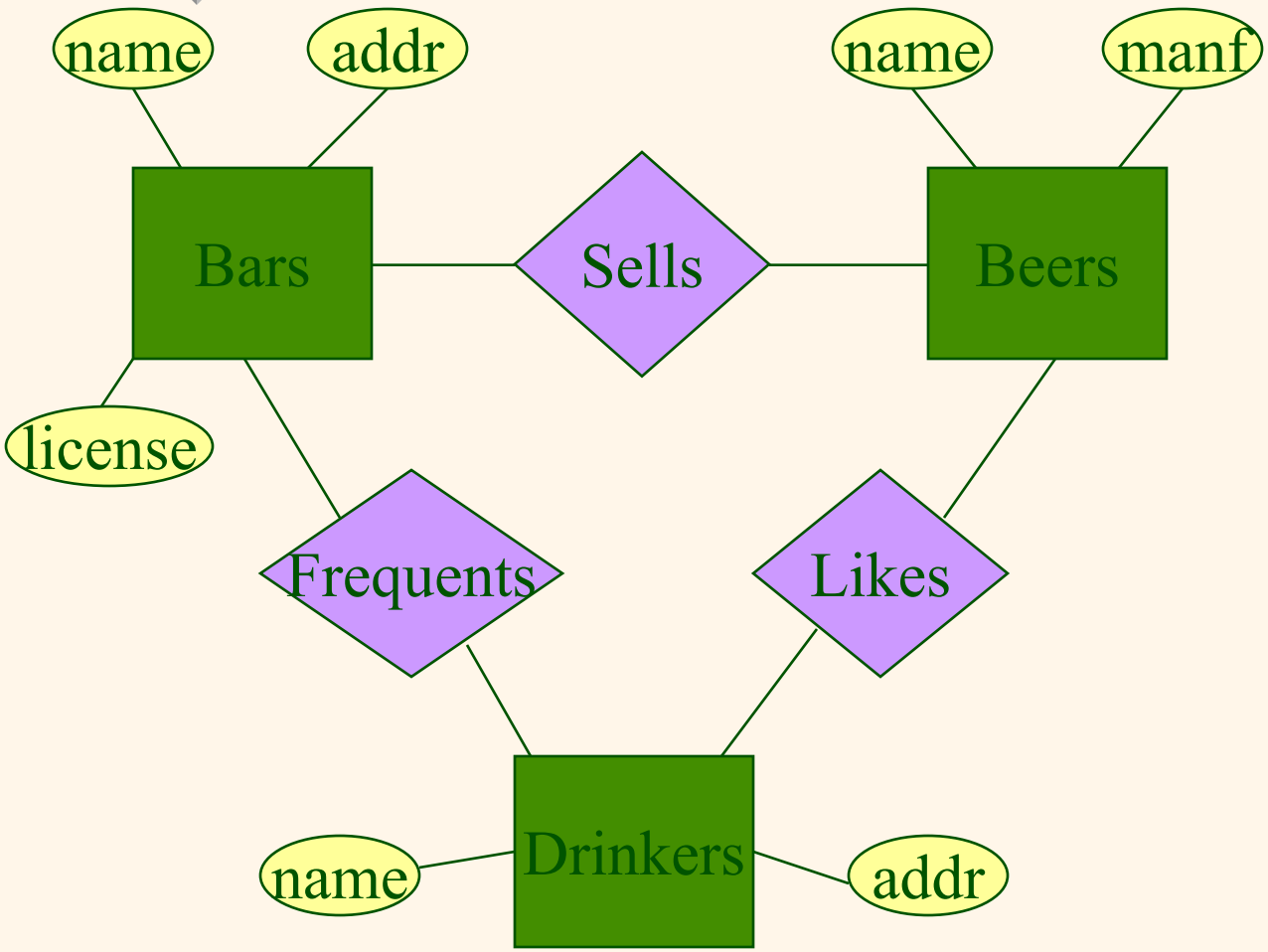
A Relationship Set Instance

bar	drinker	beer
Joe's Bar	Ann	Miller
Sue's Bar	Ann	Bud
Sue's Bar	Ann	Pete's Ale
Joe's Bar	Bob	Miller
Joe's Bar	Bob	Bud
Sue's Bar	Cathy	Pete's Ale

Recap

- Conceptual design = process of translating requirements into a formal(ish) model
- ER (Entity-Relationship) diagrams

Example



Bars sell some beers.

Drinkers like some beers.

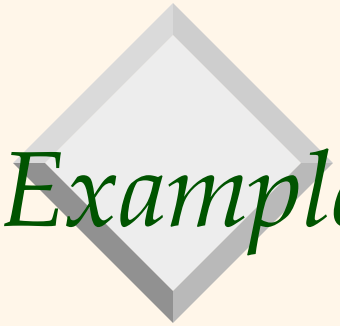
Drinkers frequent some bars.

Many-Many Relationships

- Focus: binary relationships, such as Sells between Bars and Beers.
- In a *many-many* relationship, an entity of either set can be connected to many entities of the other set.
  - a bar sells many beers; a beer is sold by many bars.

Many-One Relationships

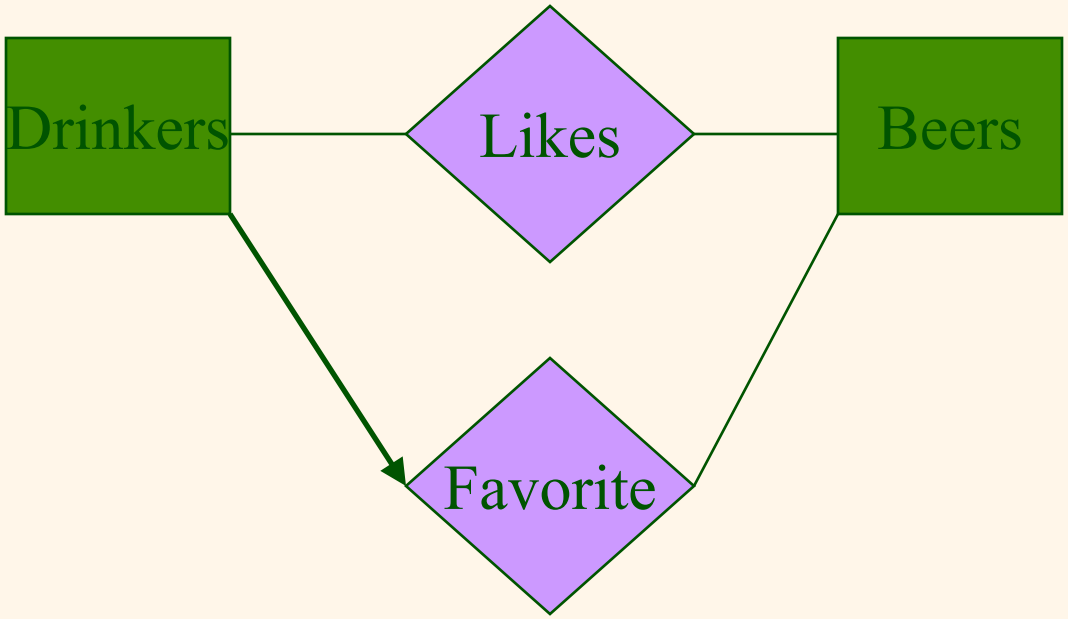
- Some binary relationships are *many -one* from one entity set to another.
- Each entity of the first set is connected to at most one entity of the second set.
- But an entity of the second set can be connected to zero, one, or many entities of the first set.



Example: Many-One Relationship

- Favorite, from Drinkers to Beers is many-one.
- A drinker has at most one favorite beer.
- But a beer can be the favorite of any number of drinkers, including zero.

Example: Many-One Relationship



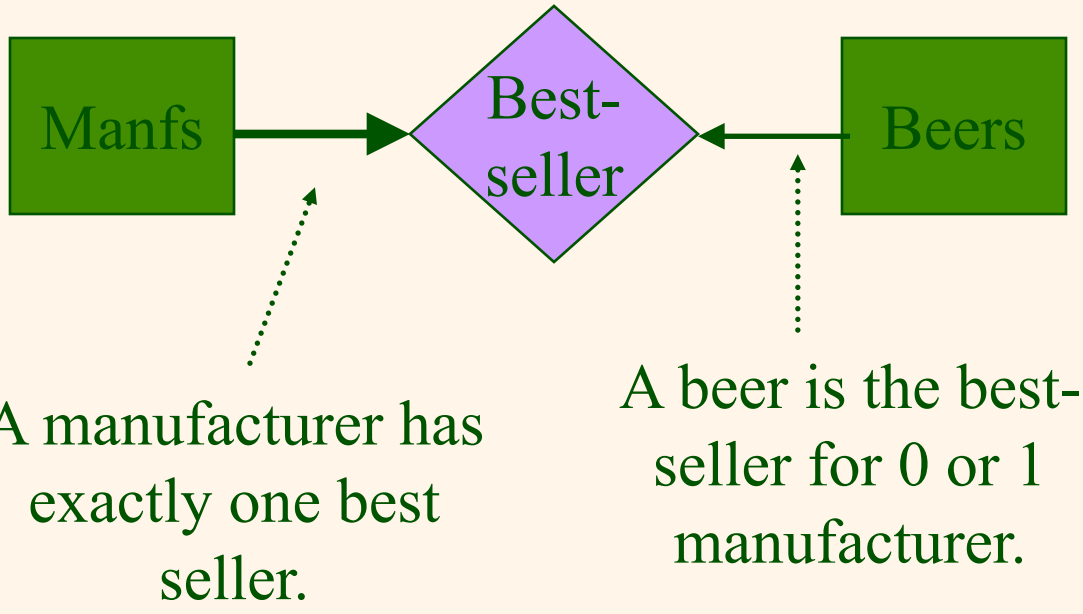
One-One Relationships

- In a *one-one* relationship, each entity of either entity set is related to at most one entity of the other set.
- Example: Relationship Best-seller between entity sets Manfs (manufacturer) and Beers.
  - A beer cannot be made by more than one manufacturer, and no manufacturer can have more than one best-seller (assume no ties).

Participation constraints

- Consider Best-seller between Manfs and Beers.
- Some beers are not the best-seller of any manufacturer
- But a beer manufacturer has to have a best-seller
  - Participation constraint – thick line

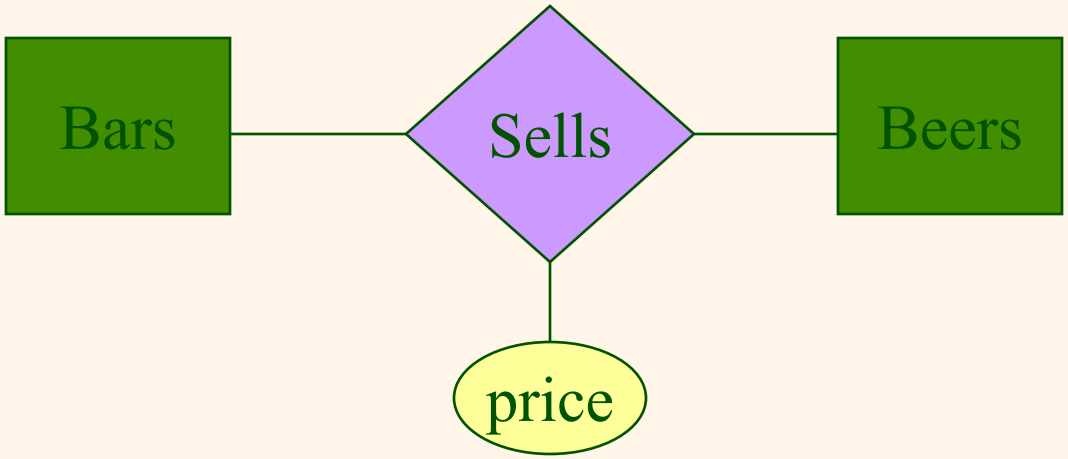
In the ER Diagram



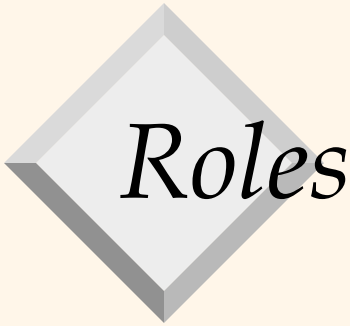
Attributes on Relationships

- Sometimes it is useful to attach an attribute to a relationship.
- Think of this attribute as a property of tuples in the relationship set.

Example: Attribute on Relationship



Price is a function of both the bar and the beer, not of one alone.

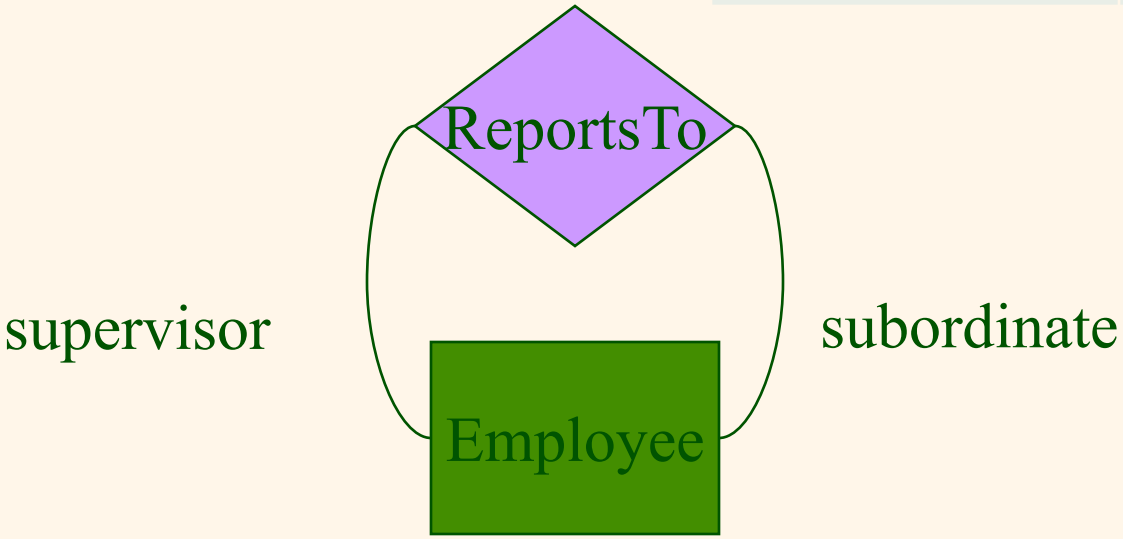


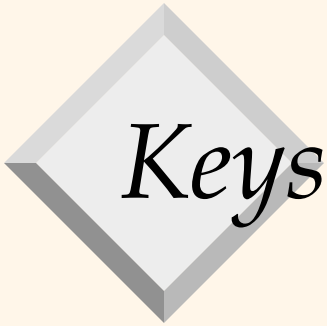
Roles

- Sometimes an entity set appears more than once in a relationship.
- Label the edges between the relationship and the entity set with names called *roles*.

Example: *Roles*

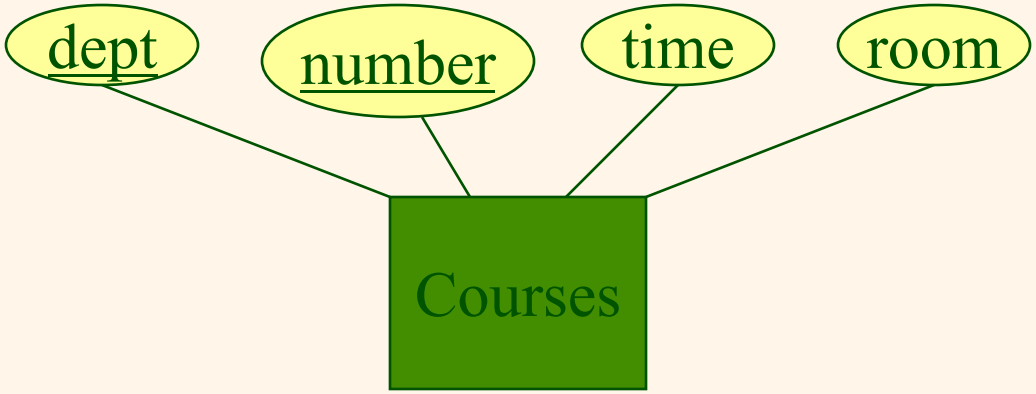
supervisor	subordinate
Bob	Ann
Joe	Sue





- A *key* is a set of attributes for one entity set such that no two entities in this set agree on all the attributes of the key.
  - It is allowed for two entities to agree on some, but not all, of the key attributes.
- We must designate a key for every entity set

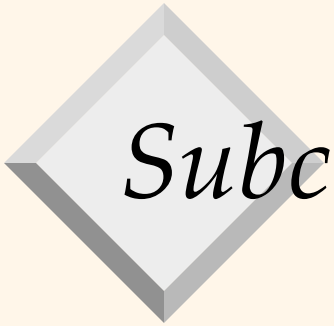
Keys in ER diagrams



• Note that time and room might also be a **candidate** key, but we must select only one **primary** key.

Keys

- In a relationship set, the keys of all the entities involved form a superkey
- This means that any specific pair (triple, etc) of entities can only appear in the instance of a relationship set once



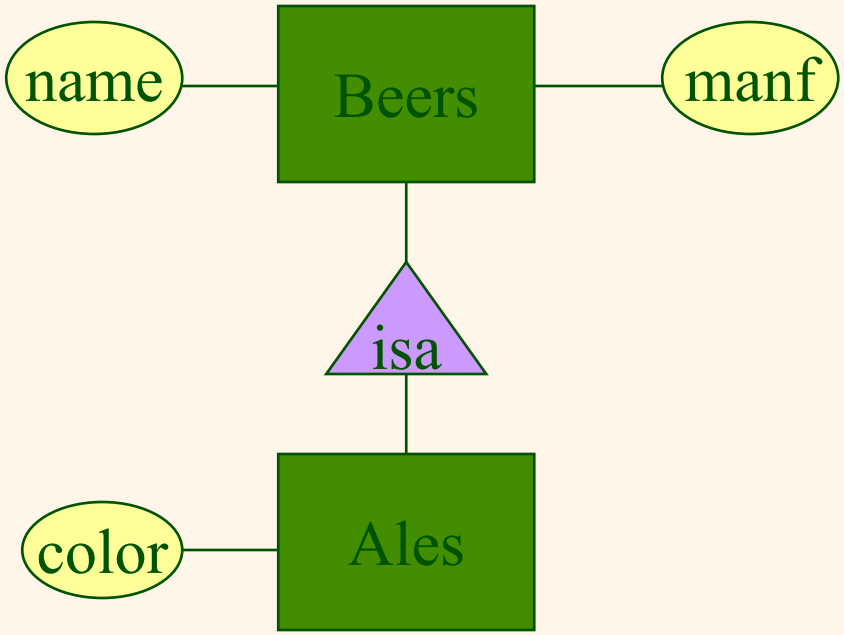
Subclasses

- *Subclasses* allow us to further distinguish beween entities within an entity set.
- Example: Ales are a kind of beer.
  - Not every beer is an ale, but some are.
  - Let us suppose that in addition to all the *properties* (attributes and relationships) of beers, ales also have the attribute color.

Subclasses in E/R Diagrams

- Assume subclasses form a tree.
  - I.e., no multiple inheritance.
- Isa triangles indicate the subclass relationship.
  - Point to the superclass.

Example

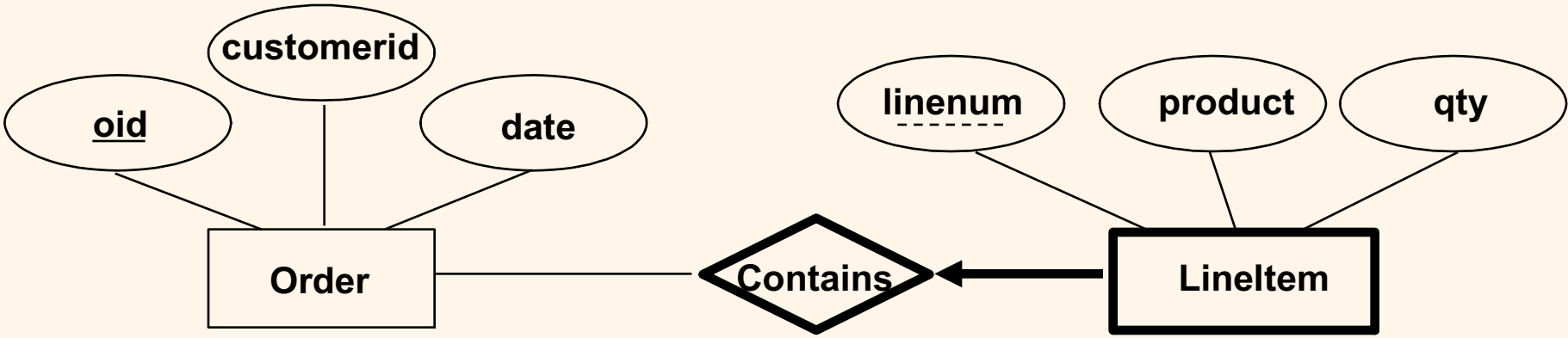


Reasons to have subclasses

- Add attributes that only make sense in some cases
  - E.g. the ale example
- "Group" entities together because they participate in the same relationship
  - E.g. Motorboats and Cars have very different attributes, but both are licensed to Owners, so want them to participate in same Licensed-To relationship.

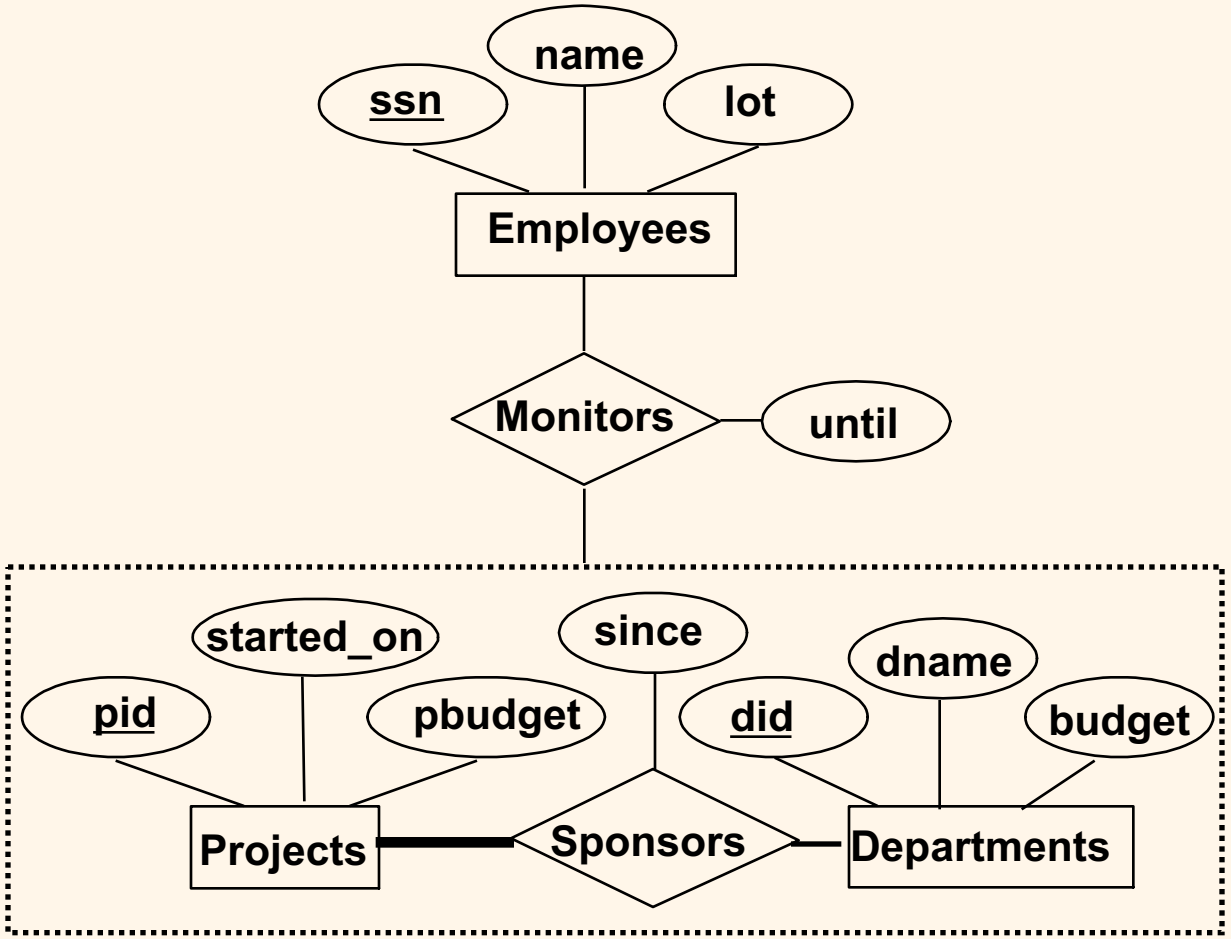
Weak Entities

- A *weak entity* can be identified uniquely only by considering the primary key of another (*owner*) entity.
  - Owner entity set and weak entity set must participate in a one-to-many relationship set (one owner, many weak entities).
  - Weak entity set must have total participation in this *identifying* relationship set.
  - linenum is a partial key for LineItem



Aggregation

- Used when we have to model a relationship involving (entitity sets and) a *relationship set*.
  - *Aggregation* allows us to treat a relationship set as an entity set for purposes of participation in (other) relationships.



☛ *Aggregation vs. ternary relationship*:

- Monitors is a distinct relationship, with a descriptive attribute.
- Also, can say that each sponsorship is monitored by at most one employee.

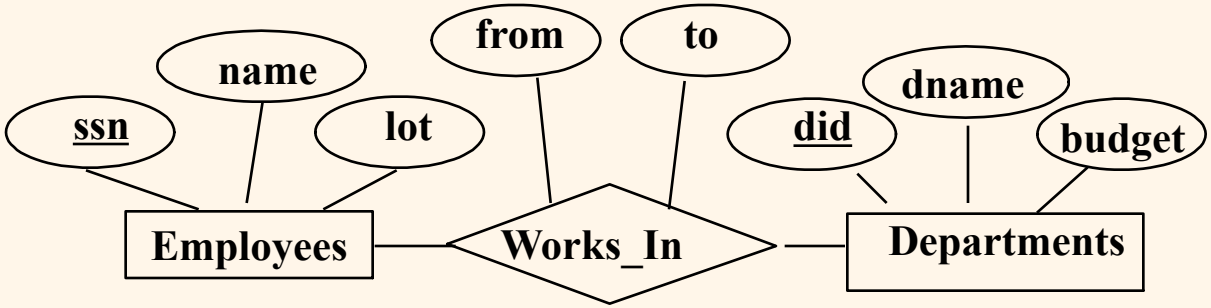
Design choices

- Should a concept be modeled as an entity or an attribute?
- Should a concept be modeled as an entity or a relationship?

Entity vs. Attribute

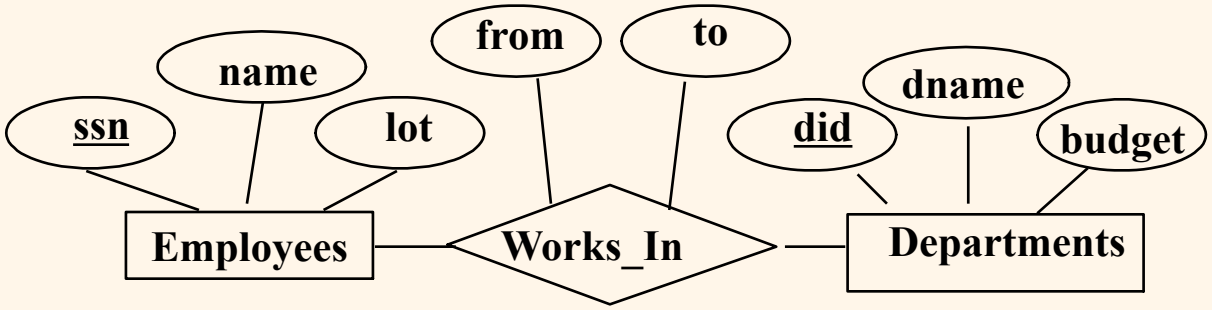
- Should address be an attribute of Employees or an entity (connected to Employees by a relationship)?
- Depends:
  - If we have several addresses per employee, address must be an entity (since attributes cannot be set-valued)
  - If the structure (city, street, etc.) is important, e.g., we want to retrieve employees in a given city, address must be modeled as an entity (since attribute values are atomic)

More design choices



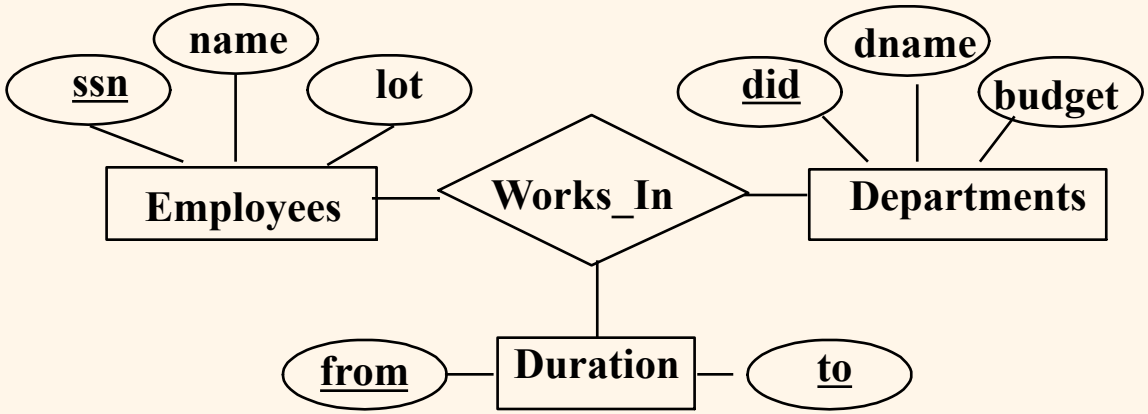
- Does not allow an employee to work in a department for two or more periods

More design choices

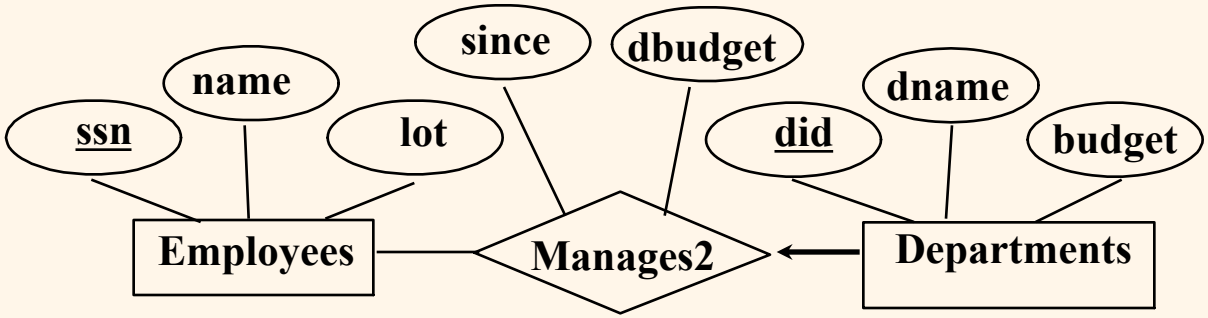


- Want to record several values of the descriptive attributes for each instance of this relationship

- Accomplished by introducing new entity set, Duration

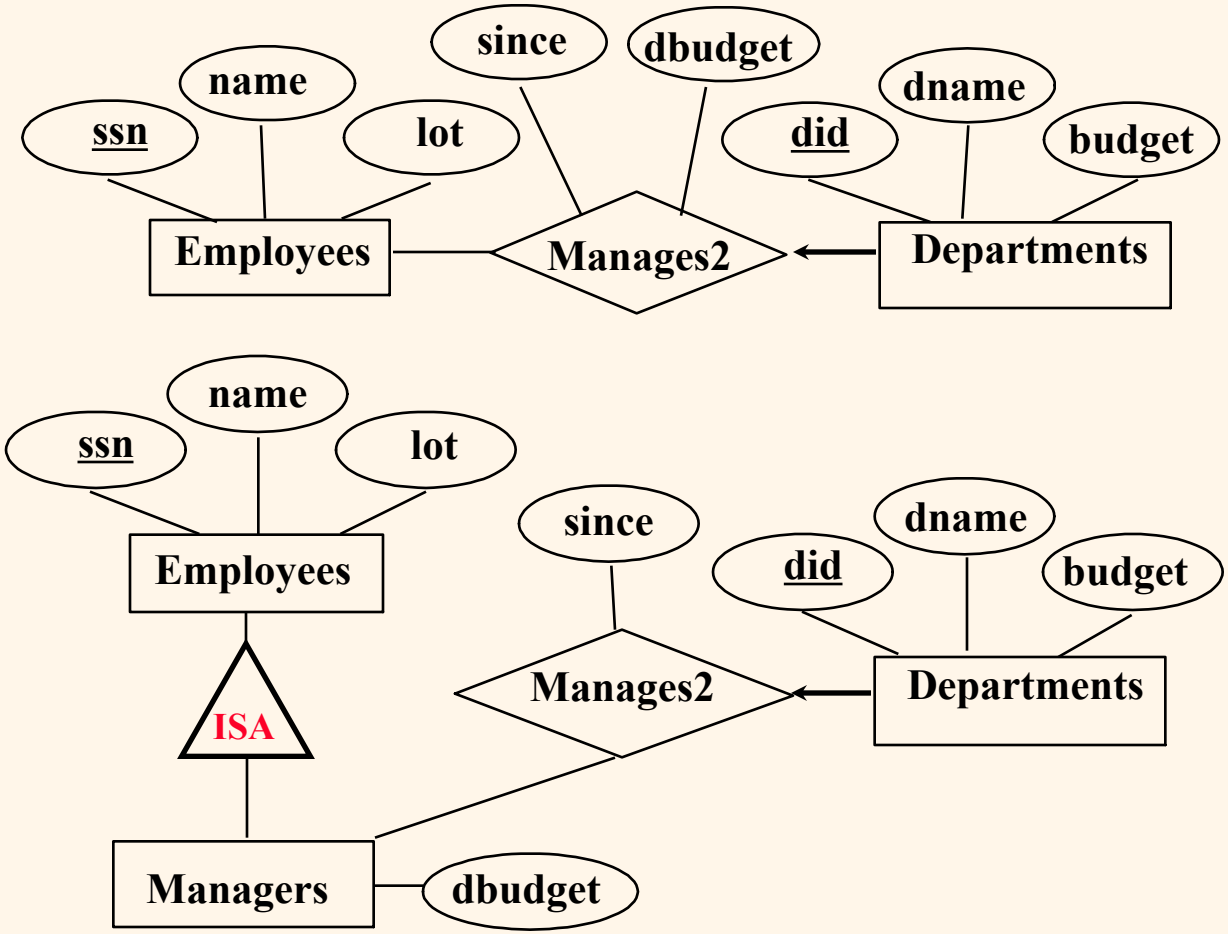


More design choices



- What if a manager gets a discretionary budget that covers *all* managed depts?
  - **Redundancy:** dbudget stored for each dept managed by manager
  - **Misleading:** Suggests dbudget associated with department-mgr combination

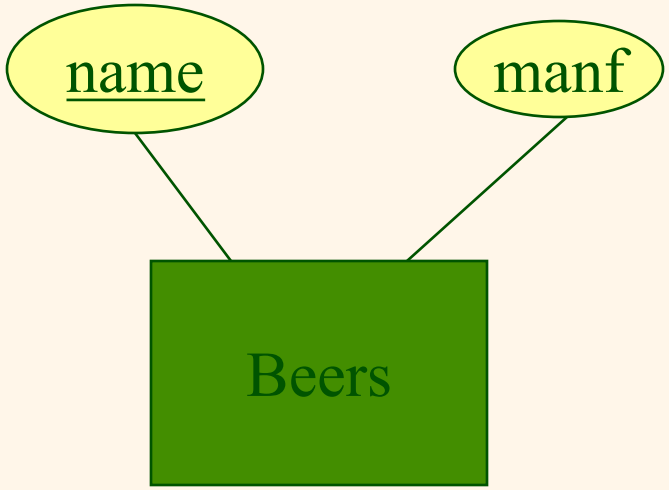
More design choices



From ER Diagrams to Relations

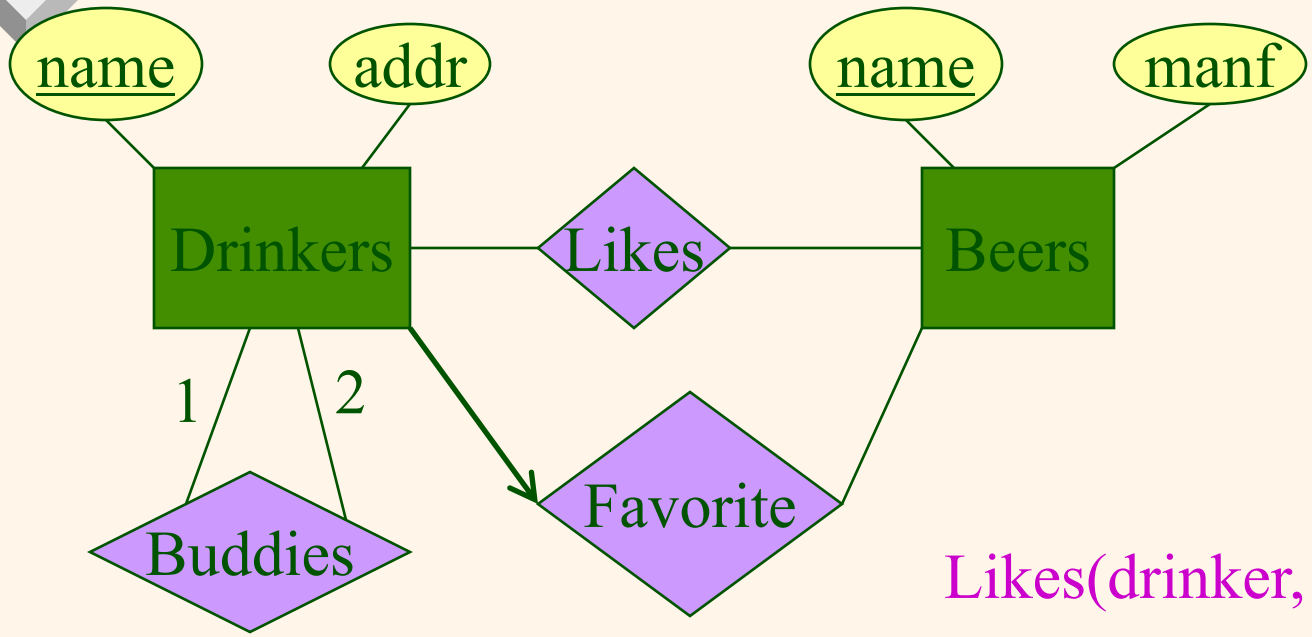
- Entity set -> relation.
  - Attributes -> attributes.
- Relationships -> relations whose attributes are only:
  - The keys of the connected entity sets.
  - Attributes of the relationship itself.
  - Though we can do better sometimes

Entity Set -> Relation



Relation: Beers(name, manf)

Relationship -> Relation



Likes(drinker, beer)

Favorite(drinker, beer)

Buddies(name1, name2)

Translating relationships to SQL

- Likes(drinker,beer)
  - Many-many
  - Optional on both sides

```
CREATE TABLE LIKES (drinker VARCHAR(15),
                    beer VARCHAR(15),
                    PRIMARY KEY (drinker, beer),
                    FOREIGN KEY drinker references DRINKERS,
                    FOREIGN KEY beer references BEERS);
```

Translating constraints to SQL

- Favorite(drinker,beer)
  - One-many
  - Optional on both sides

```
CREATE TABLE FAVORITES (drinker VARCHAR(15),
                        beer VARCHAR(15),
                        PRIMARY KEY drinker,
                        FOREIGN KEY drinker references DRINKERS,
                        FOREIGN KEY beer references BEERS);
```

Translating constraints to SQL

- Favorite(drinker,beer)
  - One-many
  - Optional on both sides
- Better: add favorite beer name to drinker table

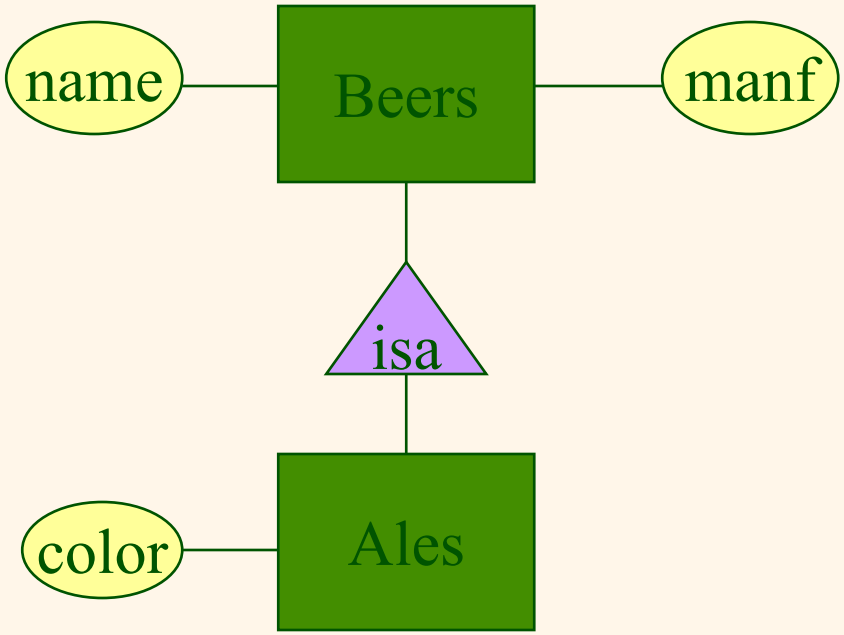
```
CREATE TABLE DRINKER(name VARCHAR(15),
                     address VARCHAR(20),
                     fav_beer VARCHAR(15),
                     PRIMARY KEY name,
                     FOREIGN KEY fav_beer references BEERS);
```

Translating constraints to SQL

- Favorite(drinker,beer)
  - One-many
  - What if every drinker must have a favorite?

```
CREATE TABLE DRINKER(name VARCHAR(15),
                     address VARCHAR(20),
                     fav_beer VARCHAR(15) NOT NULL;
                     PRIMARY KEY name,
                     FOREIGN KEY fav_beer references BEERS
                          ON DELETE NO ACTION);
```

Translating subclasses



Object-Oriented

Beers

name	manf
Bud	Anheuser-Busch

Ales

name	manf	color
Summerbrew	Pete's	dark

Good for queries like "find the color of ales made by Pete's."

ER style

name	manf
Bud	Anheuser-Busch
Summerbrew	Pete's

name	color
Summerbrew	dark

Good for queries like "find all beers (including ales) made by Pete's."

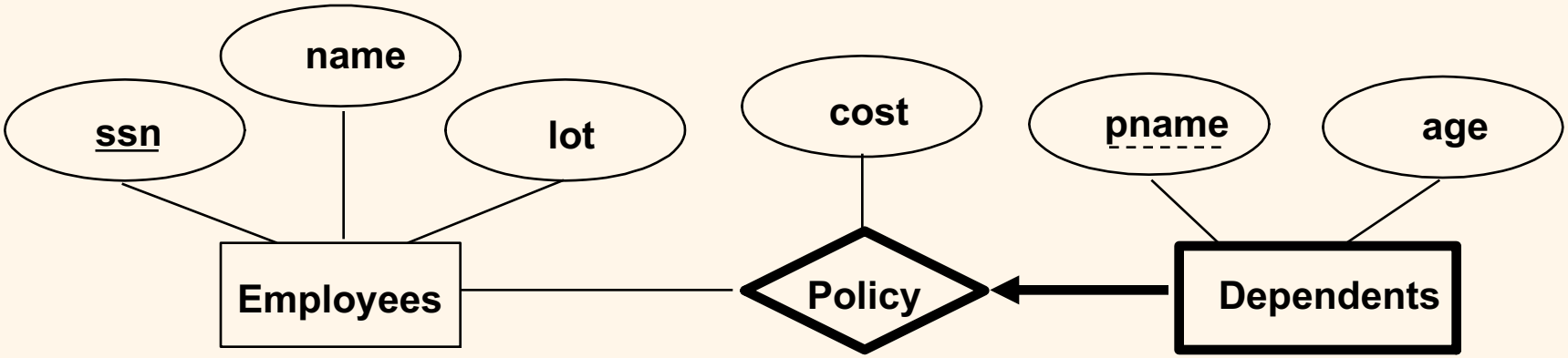
Using NULLs

name	manf	color
Bud	Anheuser-Busch	NULL
Summerbrew	Pete's	dark

Saves space unless there are *lots*
of attributes that are usually NULL.

ER to Relational (contd.)

- A *weak entity* can be identified uniquely only by considering the primary key of another (*owner*) entity.



Translating Weak Entity Sets

- Weak entity set and identifying relationship set are translated into a single table.
  - When the owner entity is deleted, all owned weak entities must also be deleted.

```
CREATE TABLE  Dep_Policy ( pname  CHAR(20),
                           age  INTEGER,
                           cost  REAL,
                           ssn  CHAR(11) NOT NULL,
                           PRIMARY KEY  (pname, ssn),
                           FOREIGN KEY  (ssn)
                                    REFERENCES Employees,
                           ON DELETE CASCADE);
```

In practice

- You might not use ER diagrams, but you will will be using some sort of diagrams as your data gets more complex
- Any nontrivial application will have a moderate number of entities and relationships
- GUI tools to help you create diagrams and translate between diagrams and relational schema (both directions)
  - E.g. MySQL Workbench

Summary so far

- Modeling your data with ER diagrams
- Translating ER diagrams to the relational model and to SQL CREATE statements
- Next: improving the relational tables you have generated to remove redundancy